Why Context Matters in a Service-based Architecture for Mobile Devices –Position Paper–

Andreas Zeidler

Darmstadt University of Technology, Department of Computer Science
Database and Distributed Systems Research Group
az@informatik.tu-darmstadt.de

1 Introduction

What would you expect when you are using an application on a mobile device? The application should perform its task as reliably and trustworthy as the same application designed for use on a desktop PC. There should be no need to reconfigure the device and the application whenever the context the device is used in, changes. Even if a mobile device is not used in steadily changing environments, say, it is used in the office for some hours, then on the train, and lastly at home in the evening, keeping track with the environment by means of *adaption* to the network, *finding services*, configuring the device to use the *right* service or use an unknown implementation of a service in a *secure* matter, is (nowadays) still a hard task. If it is left up to the user of a mobile device to adapt his or her device and its applications to the surrounding infrastructure, *ease of use* is certainly not given.

Currently used operating systems and middleware, are far from being easily usable in changing environments. To fill this gap we propose a layer of software between the operating system and the user helping to *adapt the device to the infrastructure and the infrastructure to the device* by finding surrounding services in an intelligent way, *configuring the device* to use the services found according to predefined *policies-of-use* and save users from the ever-recurring task of reconfiguration, both, on the network layer and the application layer.

Further down the road we assume that network technologies will be ubiquitous. The dream or nightmare of constantly being "online" will come true and the cost of communication will be low or even drop to zero. New business models are then centered around the notion of *services*. Usage of services will generate a revenue for the provider. From the user's point of view a new problem arises: How does a user deal with the vast variety of services popping out of the woodworks? More technical question are: How to make an appropriate rebinding of services everytime a user changes the context, by moving from one network node to another? What are the distinguishing features for using a particular service and not another? How can context be formulated?

In our research project we aim at an architecture for mobile devices providing for automated adaption of the device to the infrastructure and the infrastructure to the device, based on using context information provided by the infrastructure, the device, and the user. We call this an "adaptive service based mobile system". The enabling middleware for this project is Jini from Sun Microsystems [6].

2 Research Topics

Motivated by the introduction, the research topics presented in this paper can be grouped around a single question: *"How can someone use a mobile system without being annoyed by adapting it to changing environments?"*

Keeping the question in mind, several tasks around "context" can be identified:

Link Layer Technology. In mobile environments "connectivity" is very important. Nowadays technology supports two major communication schemas: 1:1, 1:N. Some mobile devices use infrared as link layer technology which is well suited for 1:1-communication, in the near future devices might be using Bluetooth [3], which supports a 1:N-communication pattern. Dependent on the situation communication takes place in, one schema is preferable over the other. When exchanging private data, 1:1-communication is desirable, but not when announcing the agenda of today's meeting. Two points are important to remark: (a) The communication schema *needed* is dependent on the situation or context, (b) most devices provide only one schema on the link layer. To get around this drawback imposed by the actual communication hardware, a Jini-service is under development mapping among different schemas, based on and extending the approach presented in [1]. After establishing a private communication link between the service and the device, the service transparently maps the 1:1-communication to different schemas. By now, reasonable experiences have been gathered based on establishing communication via infrared between small mobile devices, such as a 3COM Palm Pilot and the Jini infrastructure. How this applies to wireless broadcasting technology is up to be explored in the near future.

Network Transparency. For operating systems, network transparency is partly achievable by using configuration protocols, such as DHCP [4] or TFTP [5]. They provide for the assignment of network parameters, like IP addresses or the name of the Domain Name Server. On the application level more effort has to be invested for the support of roaming users, as proposed in [2], where an architecture for the employment of so called "Application Level Gateways" is presented, providing for the transparent rebinding to connectionless services, such as Mail-servers or HTTP-proxies. In short, a user can configure all network-clients, such as Web-browsers, to "localhost:port" and the application level gateway will take care of submitting the data to the appropriate server in the surrounding infrastructure. Gaining the same level of transparency for services that rely on constantly accessible network connections, like the "Network File System" (NFS), is an ongoing research topic.

Intelligent Service Rebinding. Often, the use of predefined service instances is pointless for mobile devices, e.g., all services using some sort of hardware to fulfill their tasks, such as printers or storage services needed only temporarly. In this case *context* plays a central role. When someone is scheduled to visit a customer to sign a contract and changes to it are made, he or she wishes to printout the modified contract. Using the printer at home is pointless here. More appropriate is the use of a printer near the conference room where the contract will be signed. In this example the notion of *context-awareness* is narrowed to *location-awareness* and the *finding (lookup)* of services of the

required *types*. In the more general case, several *contextual parameters* can be identified necessitating a decision which service (out of a set) is to be used in the context a person is in. Our thesis is that transforming these parameters into a context-free description language, which can be used as query-language in the surrounding infrastructure, is a suitable approach to locate services automatically. Examples might be XML or SQL. Currently we are implementing a trader architecture for Jini which allows for submitting arbitrary queries to the Jini infrastructure in order to evalute different approaches to how to find the *right* service. Often the resolution of a printer is not as important as the physical distance to the printer. Leaving open the rule for the resolution in an unexplored location (like in the example above) might be the correct query; Leaving it open when printing out holiday pictures might produce unacceptable results.

Not all decisions can be made by a subsystem of the mobile device. Some decisions are inherently "human related", e.g., which kind of food is favored on Fridays. So, additional tools are needed to formulate *policies-of-use*, describing which *selection of services* is actually presented to the user, when asked to do so. Here we propose a pragmatic approach comparable to the WYSIWYG approach found in many HTML editors: The user should not be bothered with details of the underlying description-language, but should formulate the preferences based on templates and symbolic choices. The outcome might not be perfect, but a sufficent one. How and if all of this can be done is an open research issue. By the time of this writing we have taken only first steps.

Presentation-Layer. In mobile systems the user is changing between different environments. It is not guaranteed that all environments offer the same services to the user. This is a major difference to "classical" more or less *static* networked environments with desktop PCs. In mobile scenarios there must be no need to install any special-purpose "drivers" or software, as these pieces of software are only valid in one particular environment. Furthermore, services with the same name do not always follow identical semantics. As we mentioned above, the overall goal is to guarantee that the device is functioning as good as possible independent of the context the device is used in. Hence, finding, evaluating, and presenting information about services and how to use them is of fundamental importance. Here Jini is of great help because it offers mechanisms to dynamically find services, as well as offering a strongly-typed object system extending the Java type system to the network. Furthermore, the Jini system offers *code mobility*, which means that the actual implementation and state of a service (or its proxy) is moved to the mobile device and is executed there. This concept is comparable to the "Applet" concept of Java. On the device, no *apriori* information about the type of service is needed. But this arises difficult questions: What is this particular service doing? And: How can I use this service? We addressed this questions in a laboratory course we gave recently. The outcome for the presentation layer is that the mere *type* of a service is not enough to describe it sufficently, even if it is extended to a type-hierachy. Hence, *conventions* are needed, how services are described and what information about the service is delivered. This is motivated by two reasons: If we want to find services automatically by means of *policies-of-use*, some standardized way to describe services is needed. As not all services can be allocated automatically, a description is necessary which is human readable. Moreover, applications can only use services of a type it knows about at compile-time. Luckily, this seems to be – more or less – no prob-

lem for services which can be expected to build the *core-infrastructure*, such as the hardware-centered services mentioned above. They are to be standardized and are expected to be found in every environment. Other services (e.g., value-added services) will carry along their own user interface (due to the ability of moving code). Conventions and guidelines are under development of how to integrate services into what we call a "Dynamic Service-Base for Mobile Devices", which allows for finding and displaying information about services, as well as using services with standard GUIs for different purposes, like administration and usage. The "Service-Base" is built out of Jini-services itself and knows nothing about the service-types it displays and uses.

Security. Security is inherent to all issues mentioned above and has to be enforced on all layers of an infrastructure supporting mobile systems. How this can be done is an open problem, due to the fact that a suitable *security model* for spontaneous collections of entities, as found in mobile systems, still has to be developed. To adapt known security models to infrastructures for mobile systems does not seem to be suitable. The main reasons are that they require undesirable structures like "globally unique identifiers" and globally federated key-authorities making public-keys of everyone accessible everywhere by everyone else. As a consequence, a security model must be developed based on a more locally and anonymous notion of "trust". How this applies to public networks (e.g., on trains) is questionable and beyond the main scope of this research group. Nevertheless, using "local trust models" can significantly increase security on a workgroup-level. Here, security is much easier to enforce.

3 Summary

We have given a short description of our basic assumptions and main research topics. Starting from the user's point of view, we analyzed the basic requirements for an infrastructure supporting mobility in order to give a user a good starting point to feel comfortable while using mobile devices in different contextes. We hope we made clear why and where we are using information provided by the context the mobile device is used in. The overall impression of using a reliable and trustworthy environment cannot be achieved when context is left out of consideration when designing and implementing infrastructures for the future of ubiquitous and mobile computing.

References

[1] Gerd Aschemann, Svetlana Domnitcheva, Peer Hasselmeyer, Roger Kehr, and Andreas Zeidler. A Framework for the Integration of Legacy Devices into a Jini Management Federation. In *Proceedings of the Distributed Systems: Operations and Management (DSOM99)* , pages 257–268. Springer-Verlag Berlin Heidelberg, 1999.

[2] Gerd Aschemann, Roger Kehr, and Andreas Zeidler. A Jini-based Gateway Architecture for Mobile Devices. In *Proceedings of the Java-Informations-Tage 1999 (JIT99)*, pages 203–212. Springer-Verlag Berlin Heidelberg, 1999.

[3] Bluetooth Consortium. The Bluetooth Project. `http://www.bluetooth.com/`, 1999.

[4] R. Droms. Dynamic Host Configuration Protocol (DHCP). Internet RFC 2131, March 1997.

[5] K. R. Sollins. The TFTP Protocol (Revision 2). Internet RFC 783, June 1981.

[6] Sun Microsystems Inc. *Jini Architecure Specification – Revision 1.0*, January 1999.